End-to-End Non-Factoid Question Answering with an Interactive Visualization of Neural Attention Weights

Andreas Rücklé[†] and **Iryna Gurevych**[†‡]
[†]Ubiquitous Knowledge Processing Lab (UKP)
Department of Computer Science, Technische Universität Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
www.ukp.tu-darmstadt.de

Abstract

Advanced attention mechanisms are an important part of successful neural network approaches for non-factoid answer selection because they allow the models to focus on few important segments within rather long answer texts. Analyzing attention mechanisms is thus crucial for understanding strengths and weaknesses of particular models. We present an extensible, highly modular service architecture that enables the transformation of neural network models for non-factoid answer selection into fully featured end-to-end question answering systems. The primary objective of our system is to enable researchers a way to interactively explore and compare attention-based neural networks for answer selection. Our interactive user interface helps researchers to better understand the capabilities of the different approaches and can aid qualitative analyses. The source-code of our system is publicly available.[1]

1 Introduction

Attention-based neural networks are increasingly popular because of their ability to focus on the most important segments of a given input. These models have proven to be extremely effective in many different tasks, for example neural machine translation (Luong et al., 2015; Tu et al., 2016), neural image caption generation (Xu et al., 2015), and multiple sub-tasks in question answering (Hermann et al., 2015; Tan et al., 2016; Yin et al., 2016; Andreas et al., 2016).

Attention-based neural networks are especially successful in answer selection for non-factoid questions, where approaches have to deal with complex multi-sentence texts. The objective of this task is to re-rank a list of candidate answers according to a non-factoid question, where the best-ranked candidate is selected as an answer. Models usually learn to generate dense vector representations for questions and candidates, where representations of a question and an associated correct answer should lie closely together within the vector space (Feng et al., 2015). Accordingly, the ranking score can be determined with a simple similarity metric. Attention in this scenario works by calculating weights for each individual segment in the input (attention vector), where segments with a higher weight should have a stronger impact on the resulting representation. Several approaches have been recently proposed, achieving state-of-the-art results on different datasets (Dos Santos et al., 2016; Tan et al., 2016; Wang et al., 2016).

The success of these approaches clearly shows the importance of sophisticated attention mechanisms for effective answer selection models. However, it has also been shown that attention mechanisms can introduce certain biases that negatively influence the results (Wang et al., 2016). As a consequence, the creation of better attention mechanisms can improve the overall answer selection performance. To achieve this goal, researchers are required to perform in-depth analyses and comparisons of different approaches to understand what the individual models learn and how they can be improved. Due to the lack of existing tool-support to aid this process, such analyses are complex and require substantial development effort. This important issue led us to creating an integrated solution that helps researchers to better understand the capabilities of different attention-based models and can aid qualitative analyses.

In this work, we present an extensible service architecture that can transform models for non-

[1] https://github.com/UKPLab/acl2017-non-factoid-qa

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics-System Demonstrations, pages 19–24
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-4004

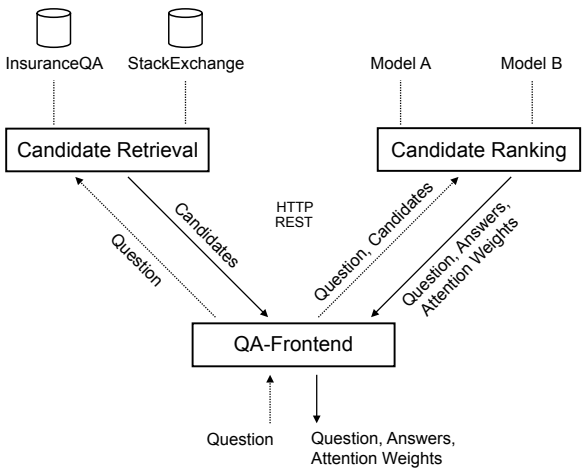

Figure 1: A high-level view on our service architecture.

factoid answer selection into fully featured end-to-end question answering systems. Our sophisticated user interface allows researchers to ask arbitrary questions while visualizing the associated attention vectors with support for both, one-way and two-way attention mechanisms. Users can explore different attention-based models at the same time and compare two attention mechanisms side-by-side within the same view. Due to the loose coupling and the strictly separated responsibilities of the components in our service architecture, our system is highly modular and can be easily extended with new datasets and new models.

2 System Overview

To transform attention-based answer selection models into end-to-end question answering systems, we rely on a service orchestration that integrates multiple independent webservices with separate responsibilities. Since all services communicate using a well-defined HTTP REST API, our system achieves strong extensibility properties. This makes it simple to replace individual services with own implementations. A high-level view on our system architecture is shown in Figure 1. For each question, we retrieve a list of candidate answers from a given dataset (candidate retrieval). We then rank these candidates with the answer selection component (candidate ranking), which integrates the attention-based neural network model that should be explored. The result contains the top-ranked answers and all associated attention weights, which enables us to interactively visualize the attention vectors in the user interface.

Our architecture is similar to the pipelined structures of earlier work in question answering that rely on a retrieval step followed by a more expensive supervised ranking approach (Surdeanu et al., 2011; Higashinaka and Isozaki, 2008). We primarily chose this architecture because it allows the user to directly relate the results of the system to the answer selection model. The use of more advanced components (e.g. query expansion or answer merging) would negate this possibility due to the added complexity.

Because all components in our extensible service architecture are loosely coupled, it is possible to use multiple candidate ranking services with different attention mechanisms at the same time. The user interface exploits this ability and allows researchers to interactively compare two models side-by-side within the same view. A screenshot of our UI is shown in Figure 2, and an example of a side-by-side comparison is available in Figure 4.

In the following sections, we describe the individual services in more detail and discuss their technical properties.

3 Candidate Retrieval

The efficient retrieval of answer candidates is a key component in our question answering approach. It allows us to narrow down the search space for more sophisticated, computationally expensive attention-based answer selection approaches in the subsequent step, and enables us to retrieve answers within seconds. We index all existing candidates of the target dataset with ElasticSearch, an open-source high-performance search engine. Our service provides a unified interface for the retrieval of answer candidates, where we query the index with the question text using BM25 as a similarity measure.

The service implementation is based on *Scala* and the *Play Framework*. Our implementation contains data readers that allow to index InsuranceQA (Feng et al., 2015) and all publicly available dumps of the StackExchange platform.[2] Researchers can easily add new datasets by implementing a single data reader class.

Analysis Enabling researchers to directly relate the results of our question answering system to

[2] https://archive.org/details/stackexchange

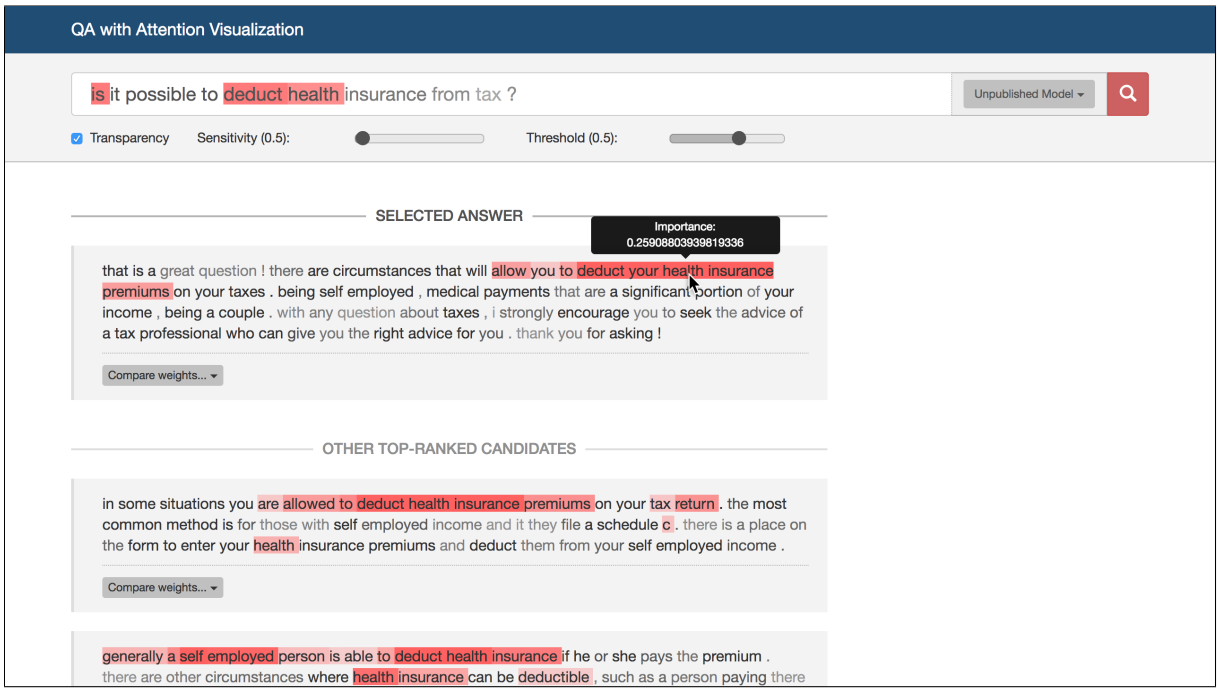

Figure 2: The user interface of our question answering system with the interactive visualization of neural attention weights. The UI includes several options to adapt the attention visualization.

the answer selection component requires the absence of major negative influences from the answer retrieval component. To analyze the potential influence, we evaluated the list of retrieved candidates (size 500) for existing questions of InsuranceQA and of different StackExchange dumps. Questions in these datasets have associated correct answers,[3] which we treat as the ground-truth that should be included in the retrieved list of candidates. Otherwise it would be impossible for the answer selection model to find the correct answer, and the results would be negatively affected. Table 1 shows the number of questions with candidate lists that include at least one ground-truth answer. Since the ratio is sufficiently high for all analyzed datasets (83% to 88%), we conclude that the chosen retrieval approach is a valid choice for our end-to-end question answering system.

Dataset	Candidate Lists with Ground-Truth
InsuranceQA (v1)	84.1% (13,200/15,687)
InsuranceQA (v2)	83.3% (14,072/16,889)
StackExchange/Travel	85.8% (13,978/16,294)
StackExchange/Cooking	88.0% (12,025/13,668)
StackExchange/Photo	83.0% (10,856/13,079)

Table 1: Performance of the retrieval service for different datasets.

4 Candidate Ranking

The candidate ranking service provides an interface to the attention-based neural network, which the researcher chose to analyze. It provides a method to rank a list of candidate answers according to a given question text. An important property is the retrieval of attention vectors from the model. These values are bundled with the top-ranked answers and are returned as a result of the service call.

Since our primary objective was to enable researchers to explore different attention-based approaches, we created a fully configurable and modular framework that includes different modules to train and evaluate answer selection models. The key properties of this framework are:

- Fully configurable with external *YAML* files.
- Dynamic instantiation and combination of configured module implementations (e.g. for the data reader and the model).
- Highly extensible: researchers can integrate new (TensorFlow) models by implementing a single class.
- Seamless integration with a webapplication that implements the service interface.

[3] For StackExchange, we consider all answers as correct that have a positive user voting. We only include questions with a positive user voting and at least one correct answer.

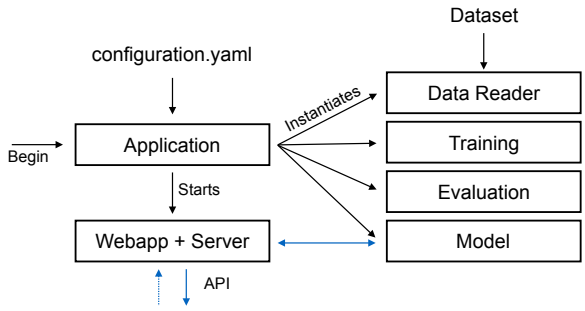

Figure 3: Our answer selection framework and candidate ranking service.

Our framework implementation is based on *Python* and relies on *TensorFlow* for the neural network components. It uses *Flask* for the service implementation.

A high-level view on the framework structure is shown in Figure 3. A particularly important property is the dynamic instantiation and combination of module implementations. A central configuration file is used to define all necessary options that enable to train and evaluate neural networks within our framework. An excerpt of such configuration is shown in Listing 1. The first four lines describe the module import paths of the desired implementations. Our framework dynamically loads and instantiates the configured modules and uses them to perform the training procedure. The remaining lines define specific configuration options to reference resource paths or to set specific neural network settings. This modular structure enables a high flexibility and provides a way to freely combine different models, training procedures, and data readers.

Additionally, our framework is capable of starting a seamlessly integrated webserver that uses a configured model to rank candidate answers. Since model states can be saved, it is possible to load pretrained models to avoid a lengthy training process.

5 QA-Frontend and User Interface

The central part of our proposed system is the QA-Frontend. This component coordinates the other services and combines them into a fully functional question answering system. Since our primary goal was to provide a way to explore and compare attention-based models, we especially focused on the user interface. Our UI fulfills the following requirements:

```
1  data-module: data.insuranceqa.v2
2  model-module: model.ap_lstm
3  training-module: training.dynamic
4  evaluation-module: evaluation.default
5
6  data:
7    map_oov: true
8    embeddings: data/glove.6B.100d.txt
9    insuranceqa: data/insuranceQA
10   ...
11
12 model:
13   lstm_cell_size: 141
14   margin: 0.2
15   trainable_embeddings: true
16   ...
17
18 training:
19   negative_answers: 50
20   batchsize: 20
21   epochs: 100
22   save_folder: checkpoints/ap_lstm
23   dropout: 0.3
24   optimizer: adam
25   scorer: accuracy
26   ...
```

Listing 1: An excerpt of a *YAML* configuration file for the candidate ranking framework.

- Use a visualization for the attention vectors similar to Hermann et al. (2015) and Dos Santos et al. (2016).
- Support for both, one-way attention mechanisms (Tan et al., 2016) and two-way attention mechanisms (Dos Santos et al., 2016).
- Enable to query multiple models within the same view.
- Provide a side-by-side comparison of different attention-based models.

We implemented the user interface with modern web technologies, such as *Angular*, *TypeScript*, and *SASS*. The QA-Frontend service was implemented in *Python* with *Flask*. It is fully configurable and allows multiple candidate ranking services to be used at the same time.

A screenshot of our user interface is shown in Figure 2. In the top row, we include an input field that allows users to enter the question text. This input field also contains a dropdown menu to select the target model that should be used for the candidate ranking. This makes it possible to ask the same question for multiple models and compare the outputs to gain a better understanding of the key differences. Below this input field we offer

multiple ways to interactively change the attention visualization. In particular, we allow to change the sensitivity s and the threshold t of the visualization component. We calculate the opacity of an attention highlight o_i that corresponds to the weight w_i in position i as follows:

$$a = min\left(w_{std},\ w_{max} - w_{avg}\right) \quad (1)$$

$$o_i = \begin{cases} s \cdot \frac{w_i - w_{avg}}{a}, & if\ \ w_i \geq w_{avg} + a \cdot t \\ 0, & otherwise \end{cases} \quad (2)$$

Where w_{avg}, w_{std} and w_{max} are the average, standard deviation and maximum of all weights in the text. We use a instead of w_{std} because in rare cases it can occur that $w_{std} > w_{max} - w_{avg}$, which would lead to visualizations without fully opaque positions. These two options make it possible to adapt the attention visualization to fit the need of the analysis. For example, it is possible to only highlight the most important sections by increasing the threshold. On the other hand, it is also possible to highlight all segments that are slightly relevant by increasing the sensitivity and at the same time reducing the threshold.

When the user hovers over an answer and the target model employs a two-way attention mechanism, the question input visualizes the associated attention weights. To get a more in-depth view on the attention vectors, the user can hover over any specific word in a text to view the exact value of the associated weight. This enables numerical comparisons and helps to get an advanced understanding of the employed answer selection model.

Finally, each answer offers the option to compare the attention weights to the output of another configured model. This action enables a side-by-side comparison of different attention mechanisms and gives researchers a powerful tool to explore the advantages and disadvantages of the different approaches. A screenshot of a side-by-side visualization is shown in Figure 4. It displays two attention mechanisms that result in very different behavior. Whereas the model to the left strongly focuses on few individual words (especially in the question), the model to the right is less selective and focuses on more segments that are similar. Our user interface makes it simple to analyze such attributes in detail.

6 Conclusion

In this work, we presented a highly extensible service architecture that can transform non-factoid answer selection models into fully featured end-to-end question answering systems. Our key contribution is the simplification of in-depth analyses of attention-based models to non-factoid answer selection. We enable researchers to interactively explore and understand their models qualitatively. This can help to create more advanced attention mechanisms that achieve better answer selection results. Besides enabling the exploration of individual models, our user interface also allows researchers to compare different attention mechanisms side-by-side within the same view.

All components of our system are highly modular which allows it to be easily extended with additional functionality. For example, our modular answer retrieval component makes it simple to integrate new datasets, and our answer ranking framework allows researchers to add new models without requiring to change any other part of the application.

The source-code of all presented components as well as the user interface is publicly available. We provide a documentation for all discussed APIs.

Acknowledgements

This work has been supported by the German Research Foundation as part of the QA-EduInf project (grant GU 798/18-1 and grant RI 803/12-1). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to Compose Neural Networks for Question Answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*. pages 1545–1554. https://doi.org/10.18653/v1/N16-1181.

Cicero Dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive Pooling Networks. *arXiv preprint* https://arxiv.org/abs/1602.03609.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding*. pages 813–820. https://doi.org/10.1109/ASRU.2015.7404872.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read

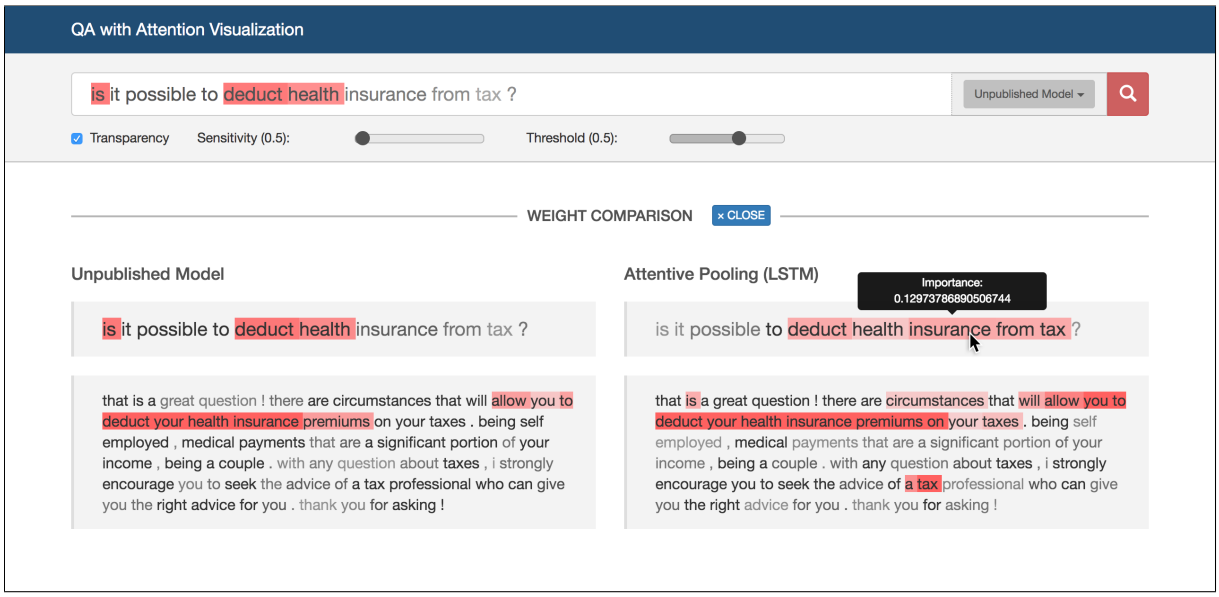

Figure 4: A side-by-side comparison of two different attention-based models. It allows the user to quickly spot the differences of the used models and can be used to better analyze their benefits and drawbacks.

and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Ryuichiro Higashinaka and Hideki Isozaki. 2008. Corpus-based question answering for why-questions. In *Proceedings of the Third International Joint Conference on Natural Language Processing*. pages 418–425. http://aclweb.org/anthology/I08-1055.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1412–1421. https://doi.org/10.18653/v1/D15-1166.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics* 37(2). http://aclweb.org/anthology/J11-2003.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 464–473. https://doi.org/10.18653/v1/P16-1044.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* pages 76–85. https://doi.org/10.1145/2856767.2856776.

Bingning Wang, Kang Liu, and Jun Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 1288–1297. https://doi.org/10.18653/v1/P16-1122.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *Proceedings of The 32nd International Conference on Machine Learning* pages 2048–2057. https://doi.org/10.1109/72.279181.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association of Computational Linguistics* 4:259–272. http://aclweb.org/anthology/Q16-1019.